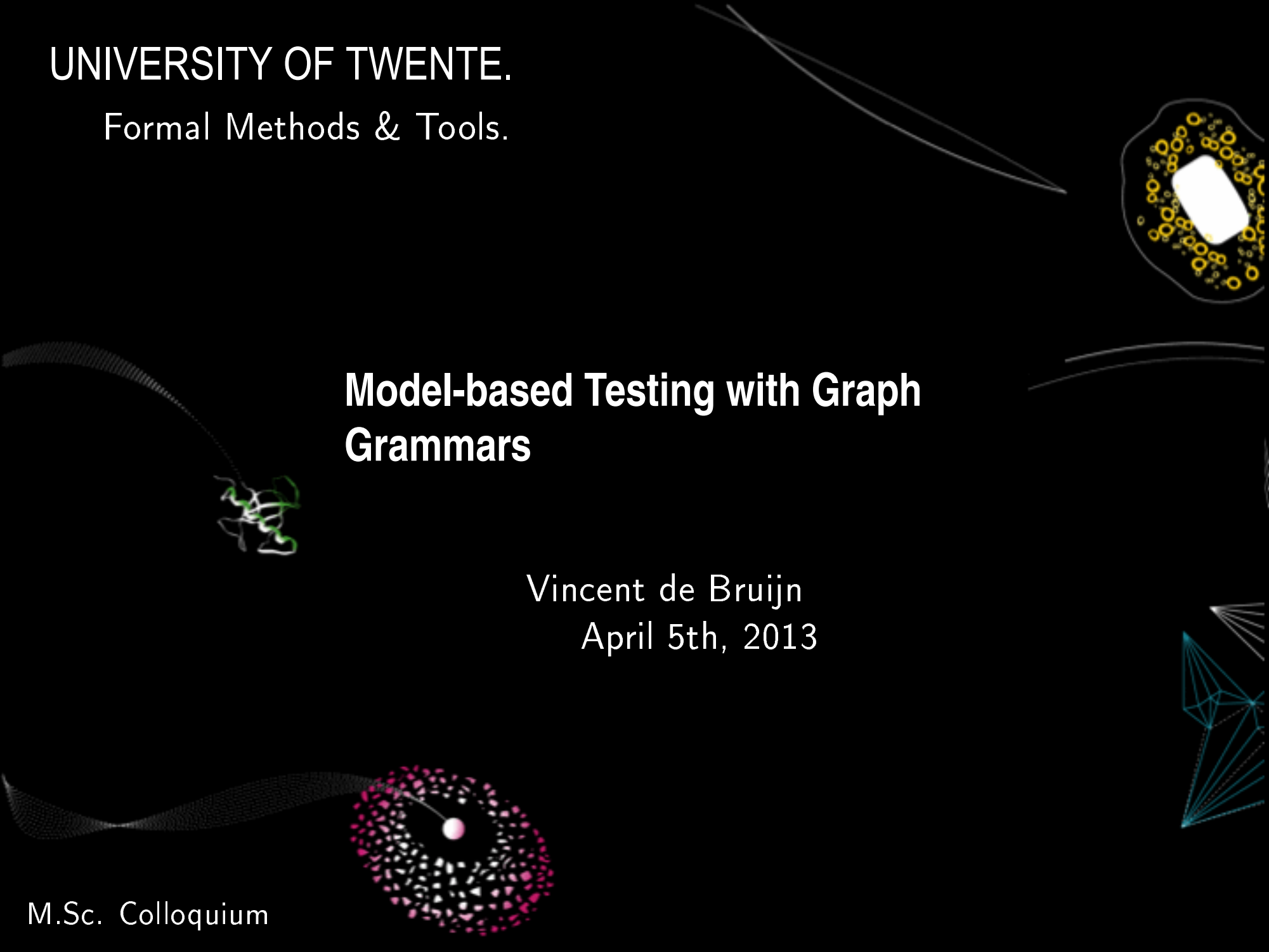
UNIVERSITY OF TWENTE.

Formal Methods & Tools.



Model-based Testing with Graph Grammars

Vincent de Bruijn

April 5th, 2013

M.Sc. Colloquium

Contents

Testing (1/3)

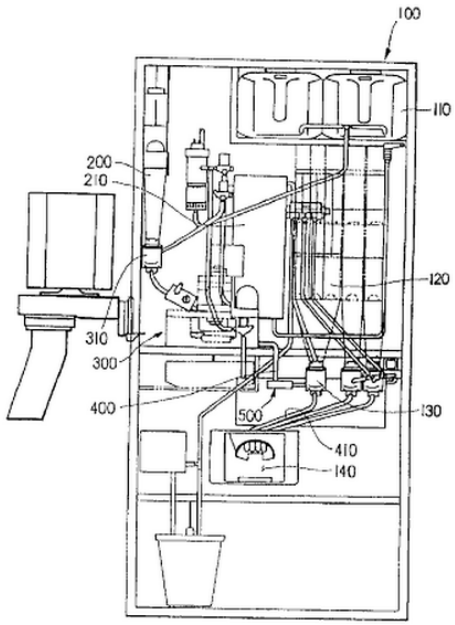
- Why do we test?
  - Products have requirements
  - Software implementation should uphold requirements

Testing (2/3)

- Creating tests manually:
  - Error-prone
  - Time intensive

Testing (3/3)

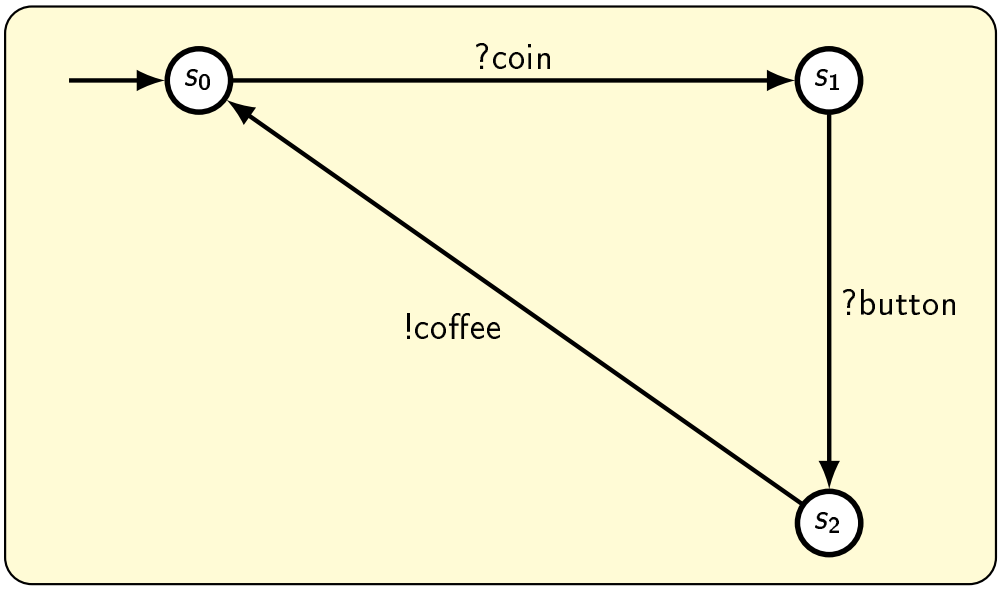
- Solution
  - Create ‘model‘ from the requirements
  - Generate tests automatically using model



Models (1/2)

Model

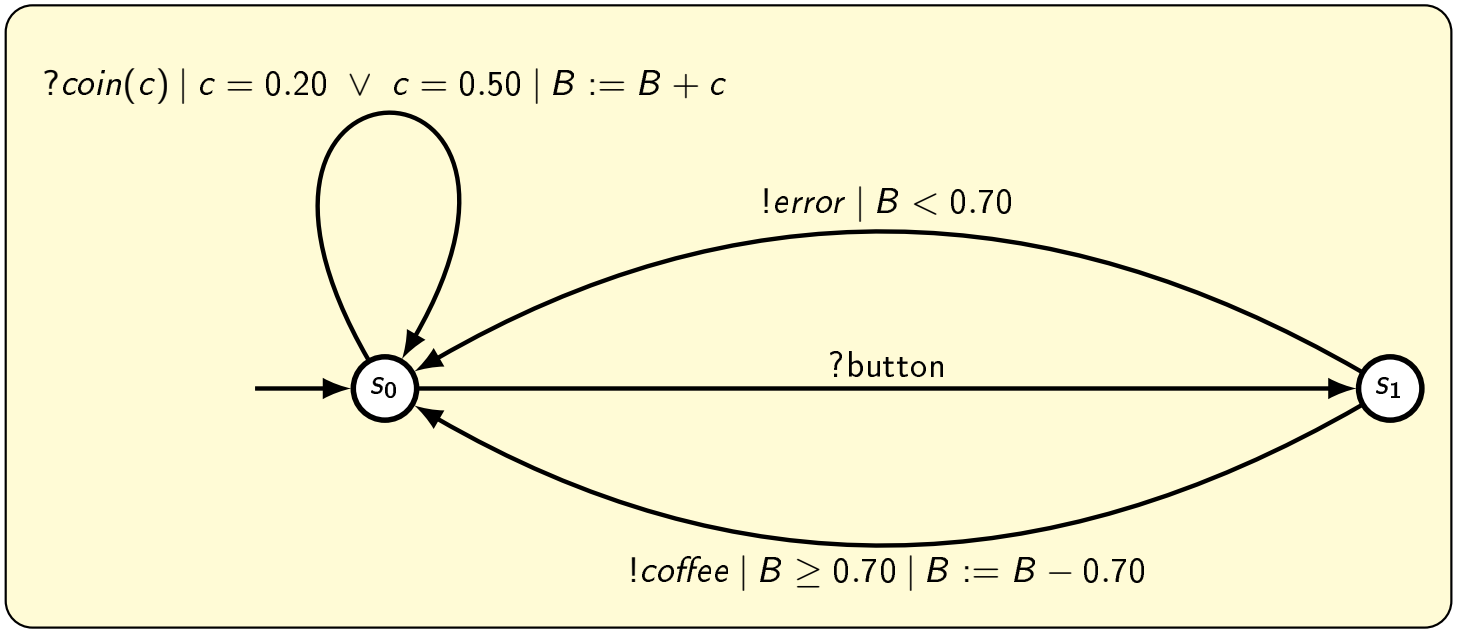
- An abstract representation of the behavior of a system



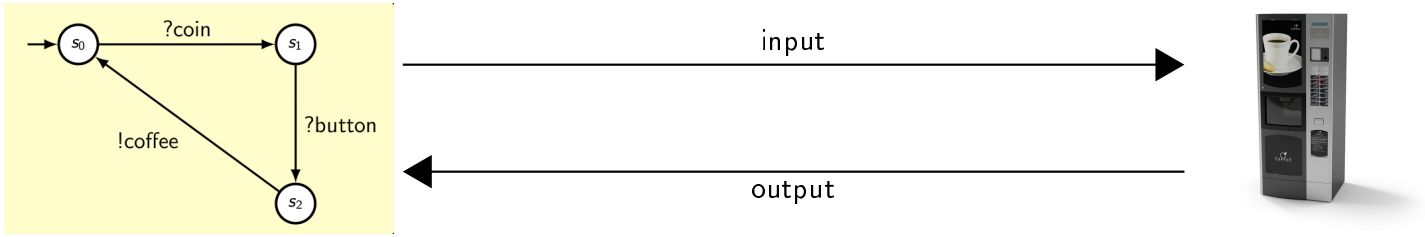
Models (2/2)

Symbolic Transition System (STS)

- Transition system with *variables*, *constraints* and *updates*

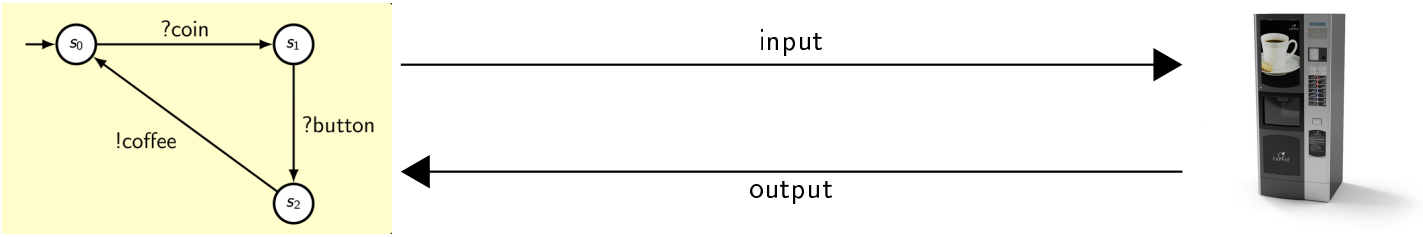


Model-based Testing (1/1)



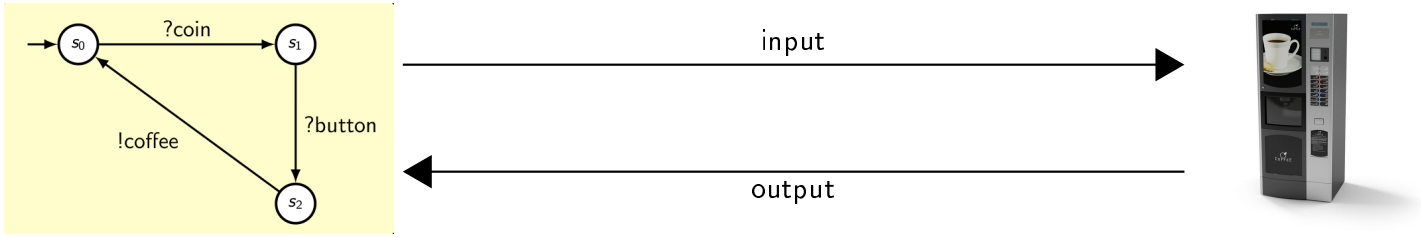
1. Take possible input from model

Model-based Testing (1/1)



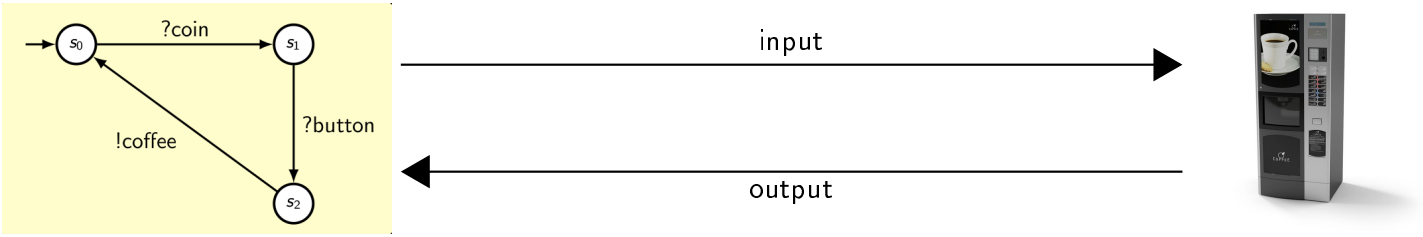
1. Take possible input from model
2. Apply input to the System Under Test (SUT)

Model-based Testing (1/1)



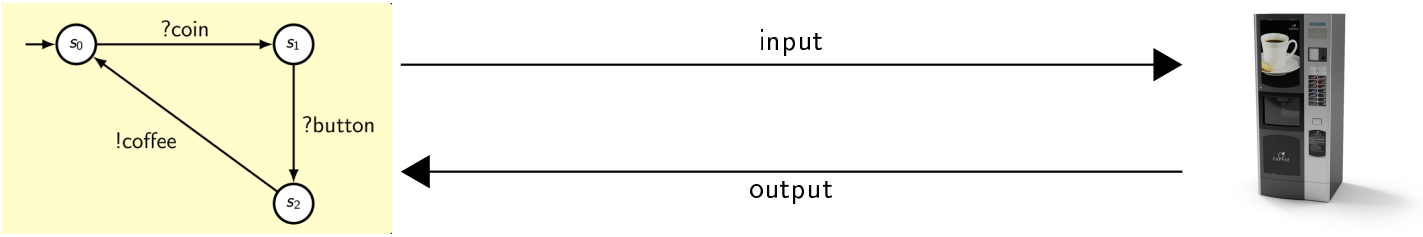
1. Take possible input from model
2. Apply input to the System Under Test (SUT)
3. Observe output

Model-based Testing (1/1)



1. Take possible input from model
2. Apply input to the System Under Test (SUT)
3. Observe output
4. Check if output is according to model

Model-based Testing (1/1)

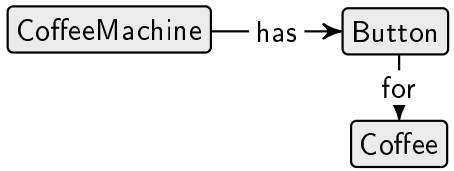


1. Take possible input from model
2. Apply input to the System Under Test (SUT)
3. Observe output
4. Check if output is according to model
5. Notify tester whether test passed or failed

Graph Grammars (1/6)

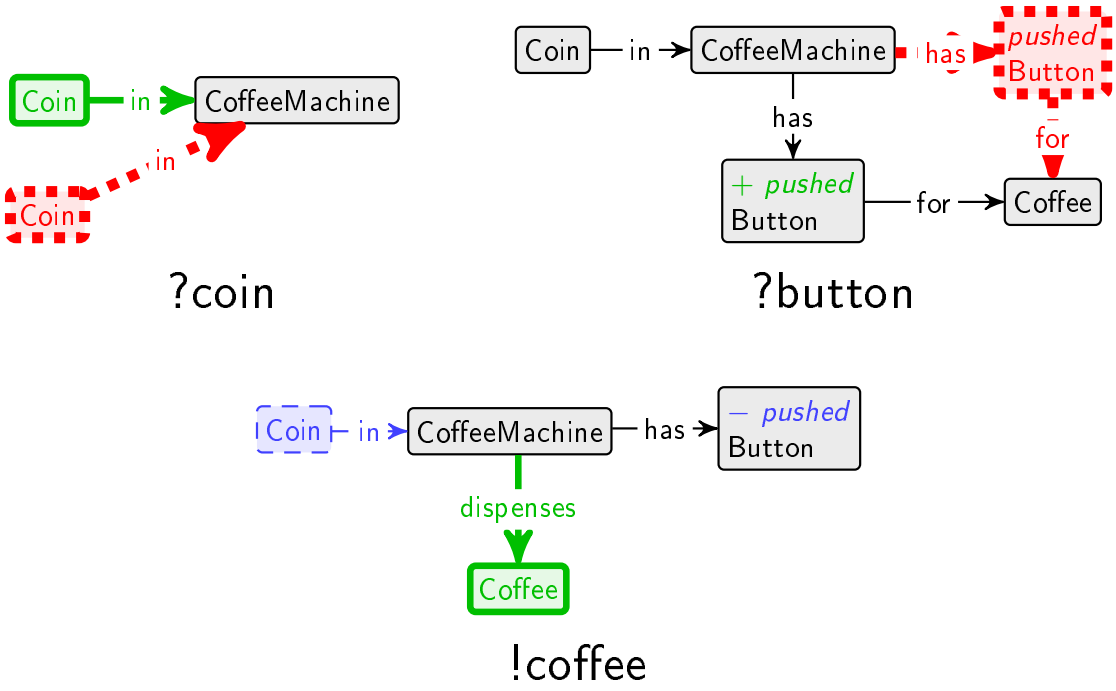
- Graphs represent system states
- Graph rules represent behavior of the system

coffee machine



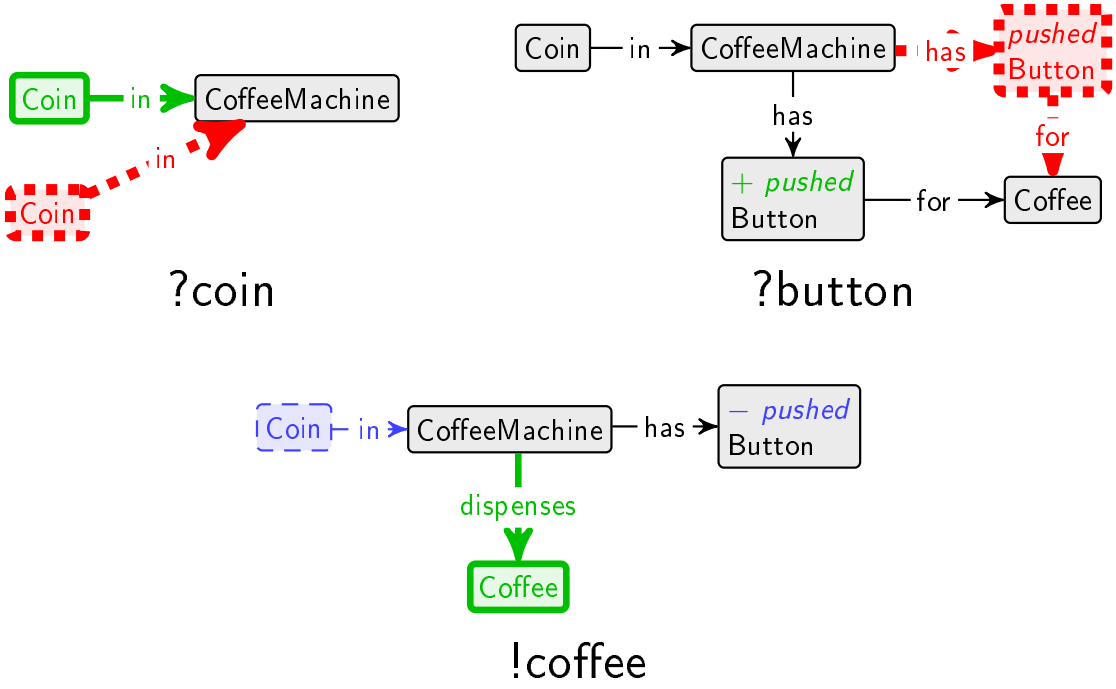
graph

Graph Grammars (2/6)



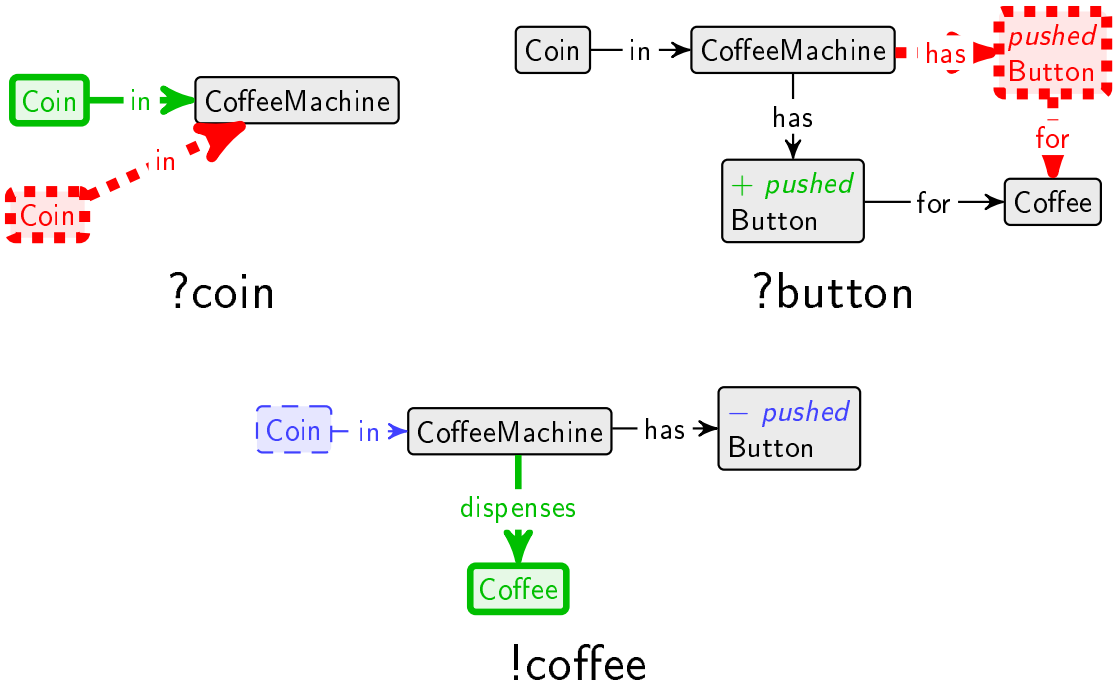
- Black and blue parts have to be present in graph

Graph Grammars (2/6)



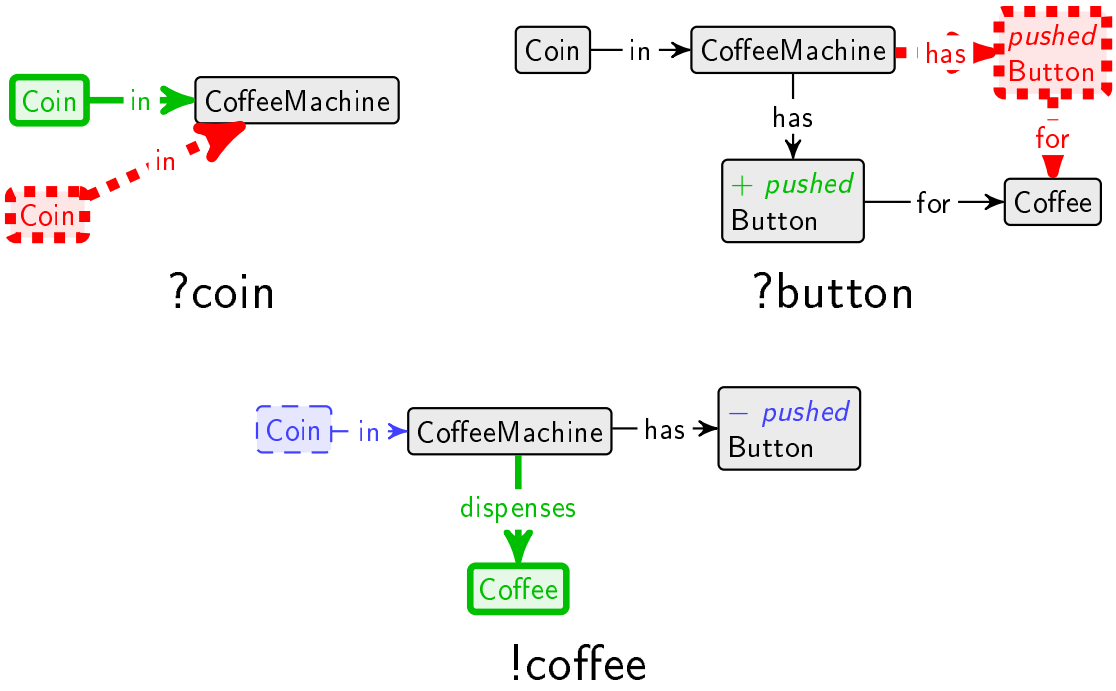
- Black and blue parts have to be present in graph
- Red parts may not be present in graph

Graph Grammars (2/6)



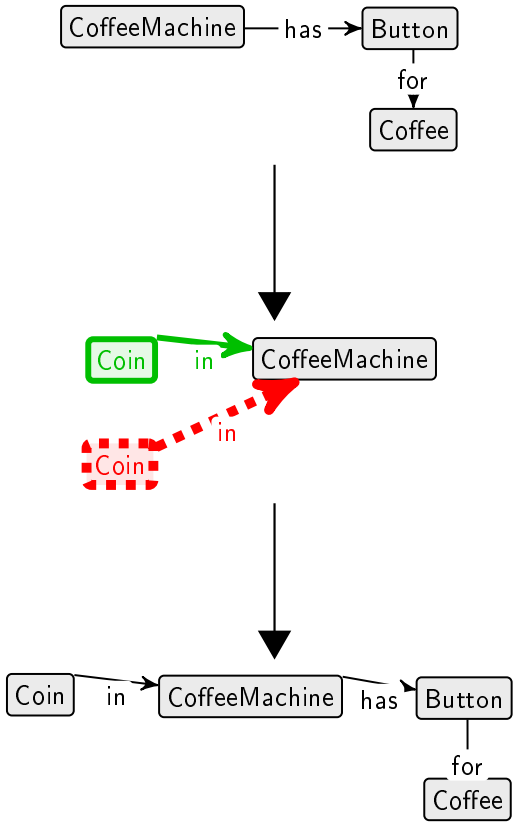
- Black and blue parts have to be present in graph
- Red parts may not be present in graph
- Blue is erased from graph

Graph Grammars (2/6)

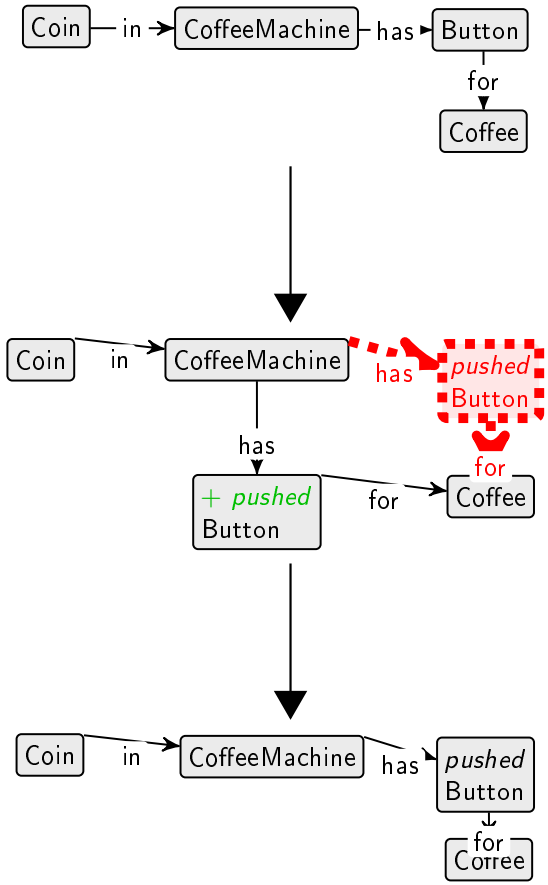


- Black and blue parts have to be present in graph
- Red parts may not be present in graph
- Blue is erased from graph
- Green is added to graph

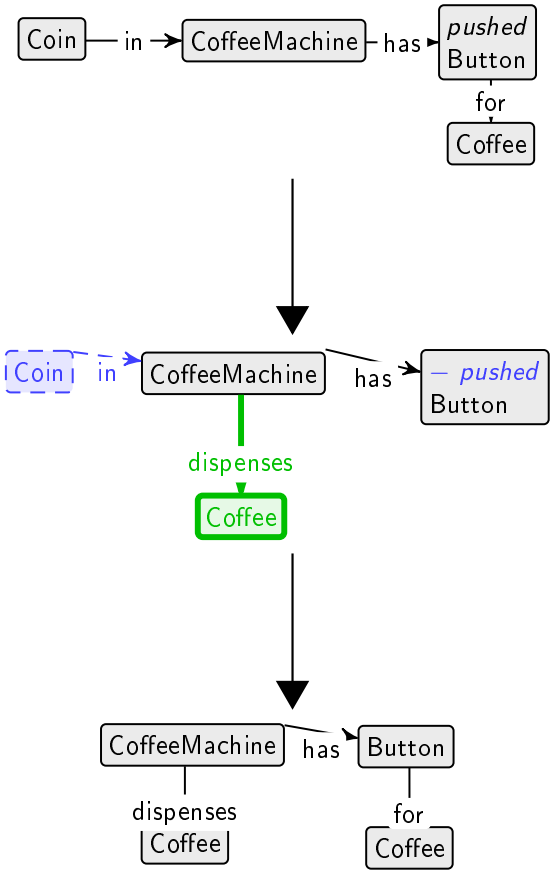
Graph Grammars (3/6)



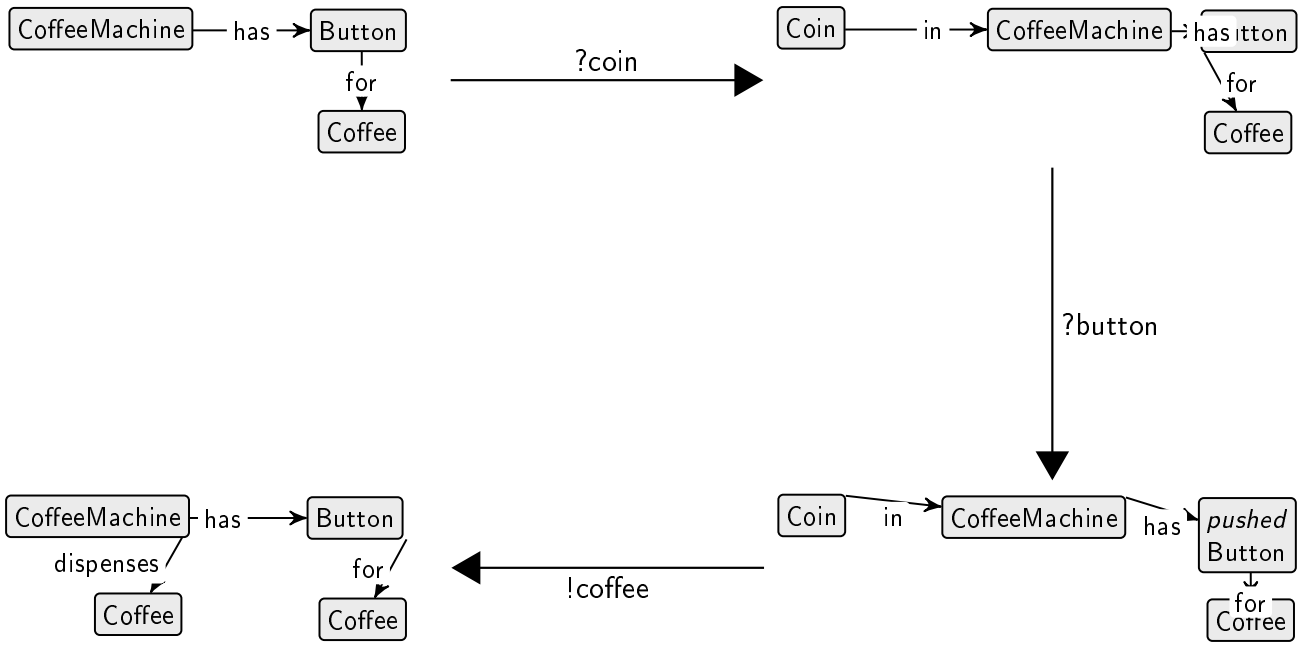
Graph Grammars (4/6)



Graph Grammars (5/6)

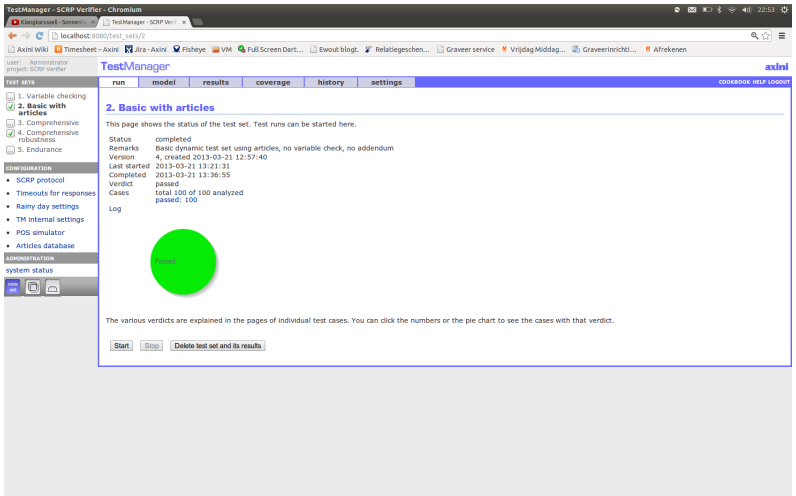


Graph Grammars (6/6)

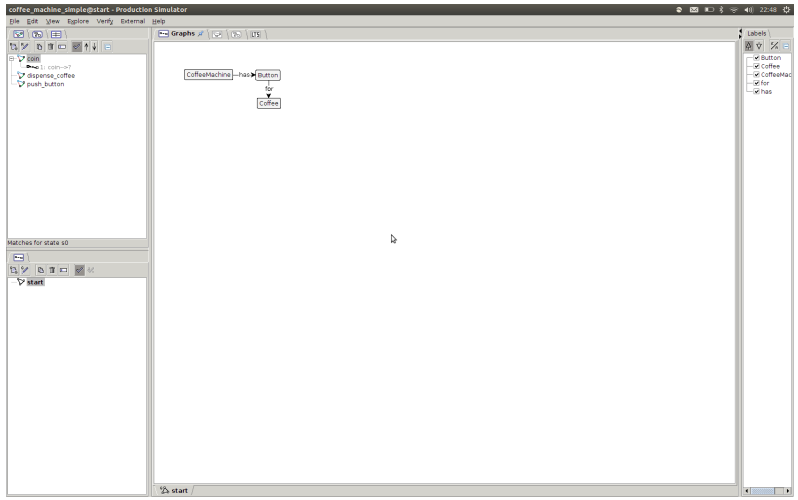


Tools or how I got to do this research

- Axini Test Manager (ATM) (uses STSs)
- GRaphs for Object-Oriented VErification (GROOVE) (uses Graph Grammars)



ATM



GROOVE

Research Goals

- Goals

Research Goals

- Goals
  - Use GROOVE and ATM to create model-based testing tool with Graph Grammars

Research Goals

- Goals
  - Use GROOVE and ATM to create model-based testing tool with Graph Grammars
  - Validate this tool using case studies

Research Goals

- Goals
  - Use GROOVE and ATM to create model-based testing tool with Graph Grammars
  - Validate this tool using case studies
- Motivation

Research Goals

- Goals
  - Use GROOVE and ATM to create model-based testing tool with Graph Grammars
  - Validate this tool using case studies
- Motivation
  - Graphs are well-known and often used to represent system states

Research Goals

- Goals
  - Use GROOVE and ATM to create model-based testing tool with Graph Grammars
  - Validate this tool using case studies
- Motivation
  - Graphs are well-known and often used to represent system states
  - Rules are useful for describing computations

Contents

Contents

Setup

- Make a tool GRATiS (GRoove-Axini Testing System)

Setup

- Make a tool GRATiS (GRoove-Axini Testing System)
  1. Make a model of system with Graph Grammars

Setup

- Make a tool GRATiS (GRoove-Axini Testing System)
  1. Make a model of system with Graph Grammars
  2. Generate STS from Graph Grammar

Setup

- Make a tool GRATiS (GRoove-Axini Testing System)
  1. Make a model of system with Graph Grammars
  2. Generate STS from Graph Grammar
  3. Model-based test system with STS

Contents

Case study: Coffee machine (1/9)

Requirements:

- Dispense coffee and tea

Case study: Coffee machine (1/9)

Requirements:

- Dispense coffee and tea
- Coffee costs 0.70 cts, tea costs 0.40 cts

Case study: Coffee machine (1/9)

Requirements:

- Dispense coffee and tea
- Coffee costs 0.70 cts, tea costs 0.40 cts
- 0.20 and 0.50 cent coins can be entered

Case study: Coffee machine (1/9)

Requirements:

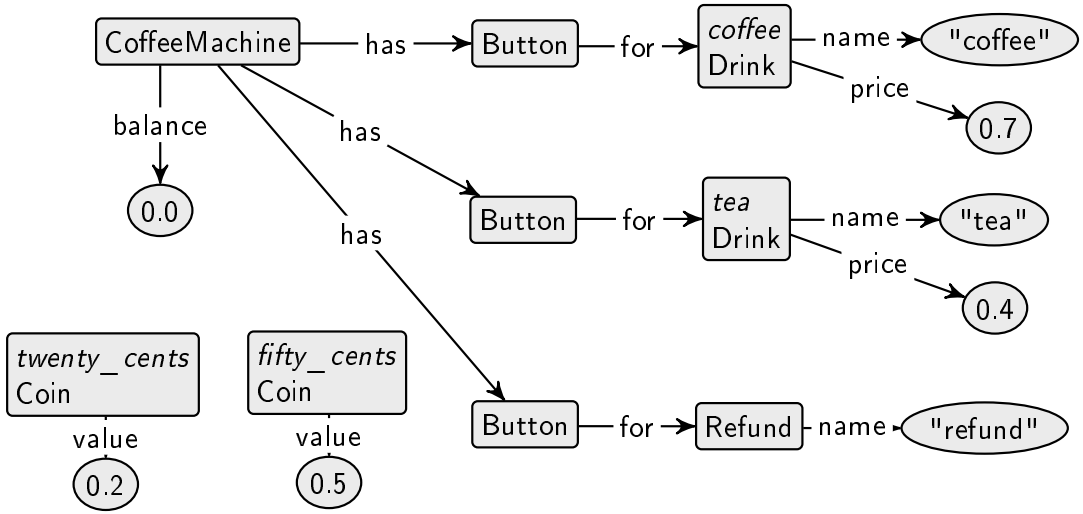
- Dispense coffee and tea
- Coffee costs 0.70 cts, tea costs 0.40 cts
- 0.20 and 0.50 cent coins can be entered
- Entered coins can be refunded

Case study: Coffee machine (1/9)

Requirements:

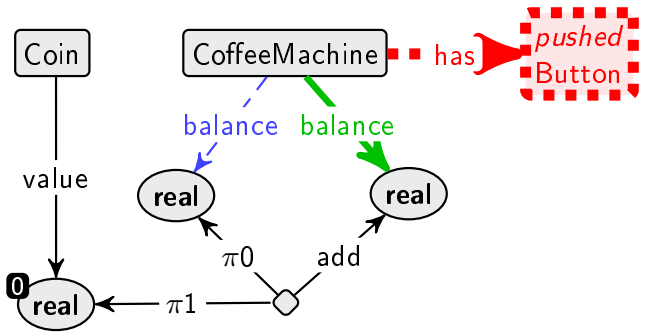
- Dispense coffee and tea
- Coffee costs 0.70 cts, tea costs 0.40 cts
- 0.20 and 0.50 cent coins can be entered
- Entered coins can be refunded
- Machine gives error when drink requested but not enough coins entered.

Case study: Coffee machine (2/9)



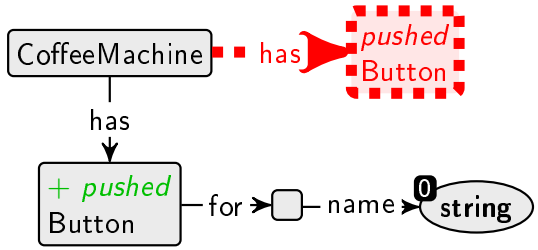
Coffee Machine start graph

Case study: Coffee machine (3/9)



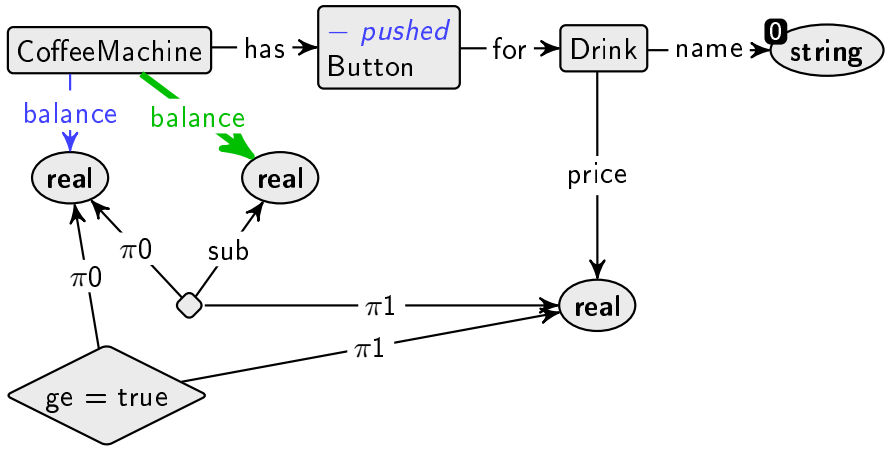
?coin(c)

Case study: Coffee machine (4/9)



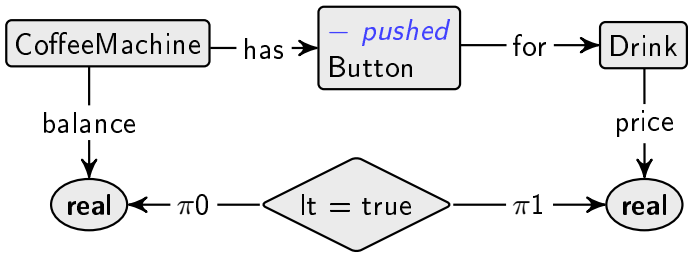
?button(b)

Case study: Coffee machine (5/9)



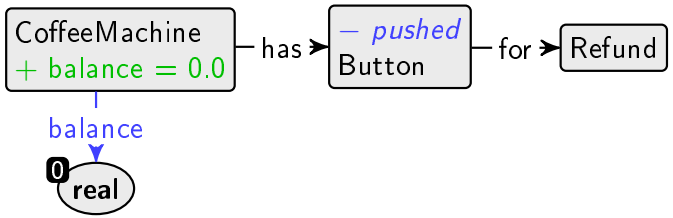
!drink(d)

Case study: Coffee machine (6/9)



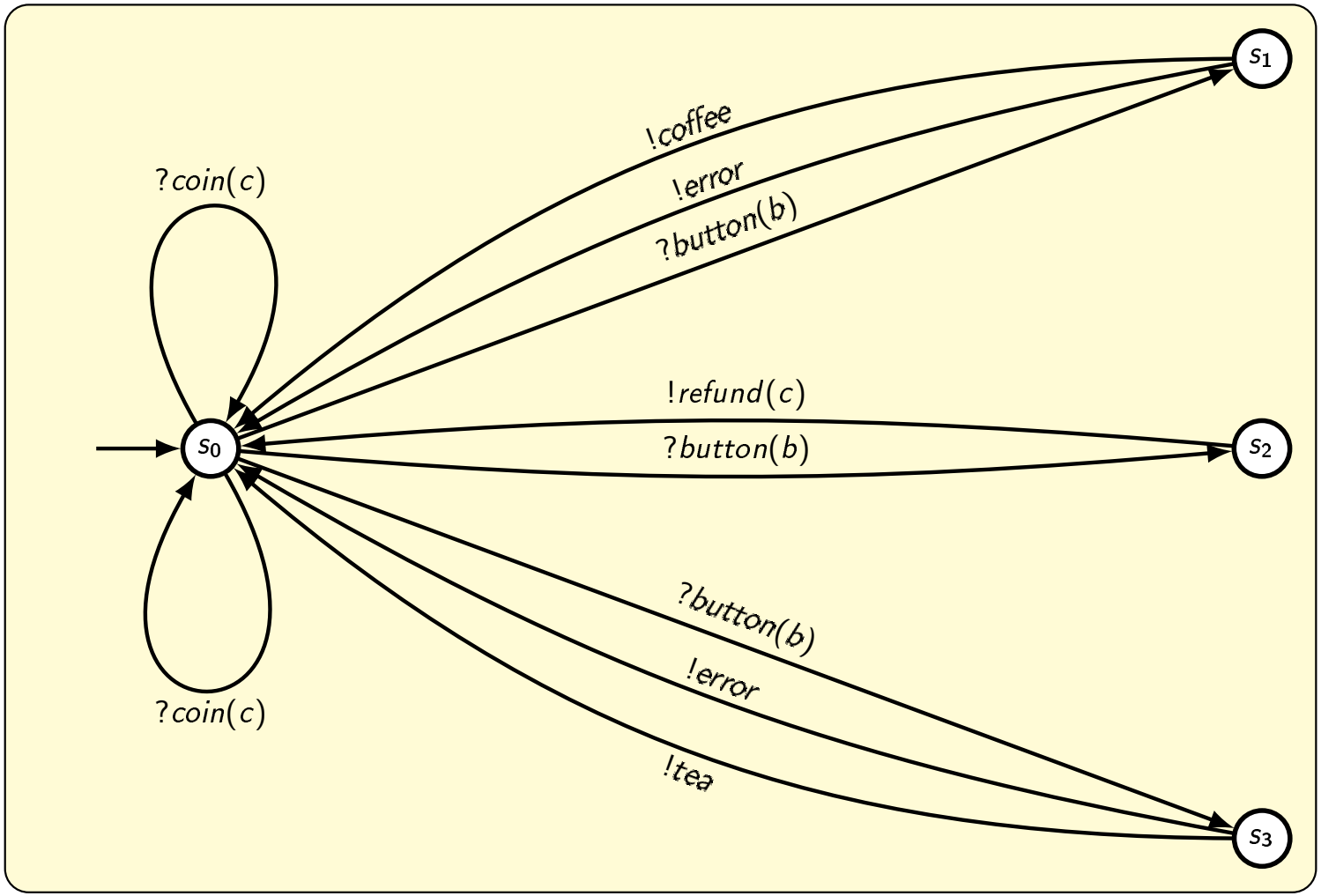
!error

Case study: Coffee machine (7/9)

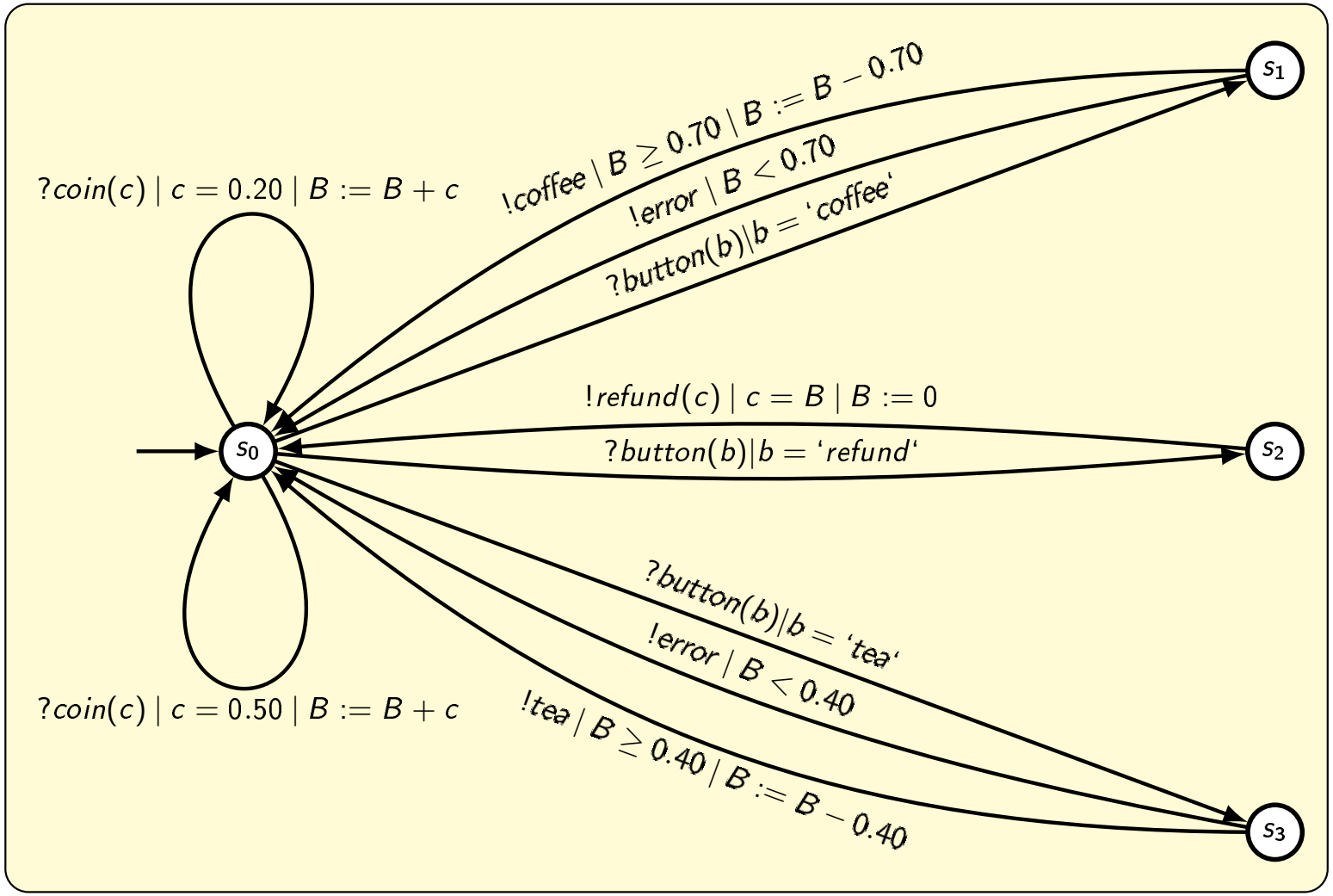


!refund(b)

Case study: Coffee machine (8/9)

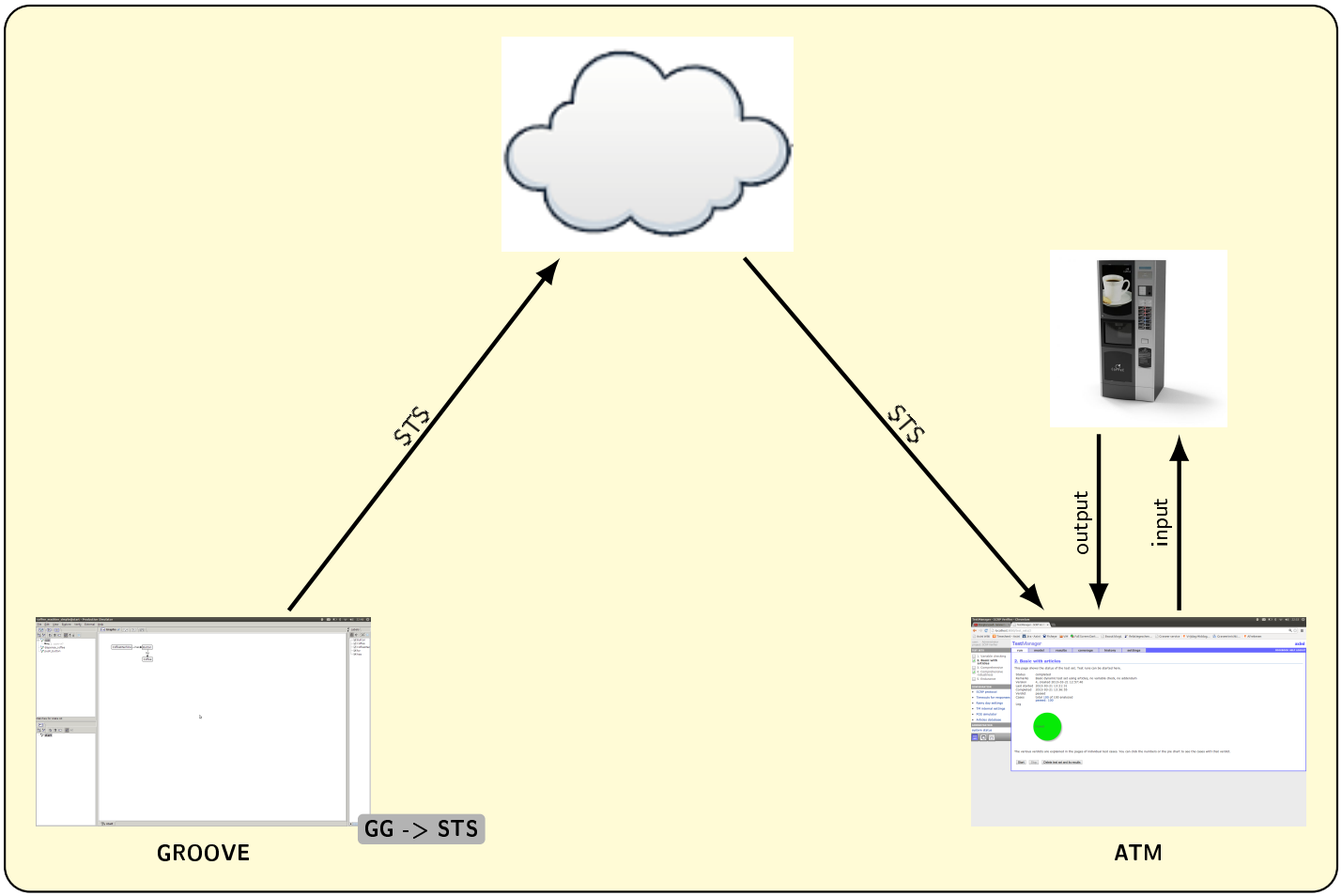


Case study: Coffee machine (9/9)



Contents

Implementation



Contents

Model Examples

- 5 example cases used:

Model Examples

- 5 example cases used:
  1. a boardgame

Model Examples

- 5 example cases used:
  1. a boardgame
  2. a puzzle

Model Examples

- 5 example cases used:
  1. a boardgame
  2. a puzzle
  3. a reservation system

Model Examples

- 5 example cases used:
  1. a boardgame
  2. a puzzle
  3. a reservation system
  4. a bar tab system

Model Examples

- 5 example cases used:
  1. a boardgame
  2. a puzzle
  3. a reservation system
  4. a bar tab system
  5. a self-checkout register

Model Examples

- 5 example cases used:
  1. a boardgame
  2. a puzzle
  3. a reservation system
  4. a bar tab system
  5. a self-checkout register
- Model examples with Graph Grammar and STS

Model Examples

- 5 example cases used:
  1. a boardgame
  2. a puzzle
  3. a reservation system
  4. a bar tab system
  5. a self-checkout register
- Model examples with Graph Grammar and STS
- Compare models

Measurements

- Performance (How fast does GRATiS make STS?)

Measurements

- Performance (How fast does GRATiS make STS?)
- Simulation (Does the STS built by GRATiS express the same behavior as the modelled STS?)

Measurements

- Performance (How fast does GRATiS make STS?)
- Simulation (Does the STS built by GRATiS express the same behavior as the modelled STS?)
- Redundancy (Is the STS built by GRATiS larger than the modelled STS?)

Measurements

- Performance (How fast does GRATiS make STS?)
- Simulation (Does the STS built by GRATiS express the same behavior as the modelled STS?)
- Redundancy (Is the STS built by GRATiS larger than the modelled STS?)
- Model complexity (Is there a difference in complexity between the STS and the Graph Grammar?)

Measurements

- Performance (How fast does GRATiS make STS?)
- Simulation (Does the STS built by GRATiS express the same behavior as the modelled STS?)
- Redundancy (Is the STS built by GRATiS larger than the modelled STS?)
- Model complexity (Is there a difference in complexity between the STS and the Graph Grammar?)
- Extendability (How easy is it to adapt both models to a hypothetical extension?)

Measurement conclusions

- Performance: less than 10 seconds for large case study

Measurement conclusions

- Performance: less than 10 seconds for large case study
- Simulation: No problems found

Measurement conclusions

- Performance: less than 10 seconds for large case study
- Simulation: No problems found
- Redundancy: Technique can create redundant STSs

Measurement conclusions

- Performance: less than 10 seconds for large case study
- Simulation: No problems found
- Redundancy: Technique can create redundant STSs
- Model complexity: Both are equally complex

Measurement conclusions

- Performance: less than 10 seconds for large case study
- Simulation: No problems found
- Redundancy: Technique can create redundant STSs
- Model complexity: Both are equally complex
- Extendability: Varying results

Contents

Conclusion

- Created method of generating STSs from Graph Grammars

Conclusion

- Created method of generating STSs from Graph Grammars
- Implemented a tool for model-based testing with Graph Grammars

Conclusion

- Created method of generating STSs from Graph Grammars
- Implemented a tool for model-based testing with Graph Grammars
- Validated the tool using case studies

Conclusion

- Created method of generating STSs from Graph Grammars
- Implemented a tool for model-based testing with Graph Grammars
- Validated the tool using case studies
- Showed modelling behavior with Graph Grammars is effective